

Linjära avbildningar

1 Inledning

Vi skall se på några geometriska transformationer; *rotation*, *skalning* och *translation*. Rotation och skalning är linjära avbildningar och kan beskrivas med standardmatriser, däremot är translation *inte* en linjär avbildning. Vi skall även se lite på ortogonal projektion och spegling.

2 Rotation, skalning och translation i $\mathbb{R}^2$

Betrakta en punkt $\mathbf{x} = (x_1, x_2)$ i $\mathbb{R}^2$. Rotation moturs med vinkeln ϕ runt origo ges av $\mathbf{T} : \mathbb{R}^2 \rightarrow \mathbb{R}^2$, där $\mathbf{T}(\mathbf{x}) = \mathbf{A}\mathbf{x}$ med standardmatrisen

$$\mathbf{A} = \begin{bmatrix} \cos(\phi) & -\sin(\phi) \\ \sin(\phi) & \cos(\phi) \end{bmatrix}$$

Motsvarande för skalning och translation blir

$$\mathbf{S}(\mathbf{x}) = \mathbf{B}\mathbf{x} = \begin{bmatrix} s_1 & 0 \\ 0 & s_2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$

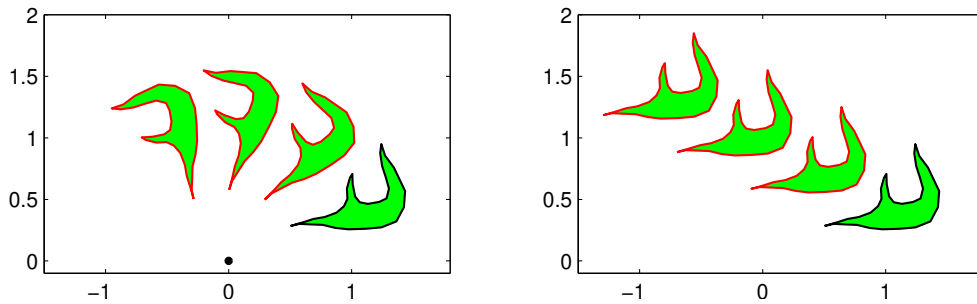
respektive

$$\mathbf{F}(\mathbf{x}) = \mathbf{x} + \mathbf{t} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} t_1 \\ t_2 \end{bmatrix}$$

För translation finns ingen standardmatris eftersom det inte är en linjär avbildning.

Vi illustrerar med MATLAB. I figuren nedan till vänster ser vi ett polygonområde med svart rand som vi roterar några gånger med vinkeln $\frac{\pi}{6}$ och ritar med röd rand. Vi tänker oss att vi redan skapat koordinater i radvektorer $\mathbf{X}$ och $\mathbf{Y}$ som beskriver det ursprungliga området.

```
fill(X,Y,'g','edgecolor','k'), hold on
axis equal, axis([-1.5 2 -0.1 2]), pause(1)
v=pi/6; A=[cos(v) -sin(v); sin(v) cos(v)];
P=[X;Y];
for i=1:3
    P=A*P;    % Varje koordinatpar roteras med vinkeln pi/6
    fill(P(1,:),P(2:,:), 'g', 'edgecolor', 'r'), pause(1)
end
plot(0,0,'ko','markersize',2) % origo
hold off
```



Till höger ser vi samma område i ursprungsläget (svart kant) samt några upprepade translationer (röd kant) med vektorn t .

```
fill(X,Y,'g','edgecolor','k'), hold on
axis equal, axis([-1.5 2 -0.1 2]), pause(1)
t=[-0.6;0.3];
P=[X;Y];
for i=1:3
    P=P+t*ones(size(X));
    fill(P(1,:),P(2,:), 'g','edgecolor','r'), pause(1)
end
hold off
```

Uppgift 1. Roter och translatera ett polygonområde ni genererar själva, t.ex. en triangel.

Innan vi går vidare är det viktigt att tänka igenom det vi gjorde: Vi antar att vi redan har två radvektorer med koordinater för polygonområde och placerar dem i en matris med $P=[X;Y]$ så att koordinaterna för punkterna ligger som kolonner.

$$P = \begin{bmatrix} x_1 & x_2 & \cdots & x_n \\ y_1 & y_2 & \cdots & y_n \end{bmatrix} = [P_1 \ P_2 \ \cdots \ P_n], \text{ med } P_i = \begin{bmatrix} x_i \\ y_i \end{bmatrix}$$

I MATLAB roterar vi med $P=A*P$ så att varje punkt roteras $\hat{P}_i = AP_i$, fast alla samtidigt.

$$\hat{P} = [\hat{P}_1 \ \hat{P}_2 \ \cdots \ \hat{P}_n] = [AP_1 \ AP_2 \ \cdots \ AP_n] = A [P_1 \ P_2 \ \cdots \ P_n] = AP$$

Vi translaterar i MATLAB med $P=P+t*\text{ones}(\text{size}(X))$ så att varje punkt translateras $\hat{P}_i = P_i + t$, fast alla samtidigt.

$$\begin{aligned} \hat{P} &= [\hat{P}_1 \ \hat{P}_2 \ \cdots \ \hat{P}_n] = [P_1 + t \ P_2 + t \ \cdots \ P_n + t] = \\ &= P + [t \ t \ \cdots \ t] = P + \begin{bmatrix} t_1 & t_1 & \cdots & t_1 \\ t_2 & t_2 & \cdots & t_2 \end{bmatrix} = P + \begin{bmatrix} t_1 \\ t_2 \end{bmatrix} [1 \ 1 \ \cdots \ 1] \end{aligned}$$

3 Rotation, skalning och translation i $\mathbb{R}^3$

Betrakta en punkt $\mathbf{x} = (x_1, x_2, x_3)$ i $\mathbb{R}^3$. Rotation med vinkeln ϕ kring x_1 -, x_2 - och x_3 -axlarna ges av $T: \mathbb{R}^3 \rightarrow \mathbb{R}^3$, där $T(\mathbf{x}) = A\mathbf{x}$ med följande respektive standardmatriser

$$A = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\phi) & -\sin(\phi) \\ 0 & \sin(\phi) & \cos(\phi) \end{bmatrix}, \quad A = \begin{bmatrix} \cos(\phi) & 0 & \sin(\phi) \\ 0 & 1 & 0 \\ -\sin(\phi) & 0 & \cos(\phi) \end{bmatrix}, \quad A = \begin{bmatrix} \cos(\phi) & -\sin(\phi) & 0 \\ \sin(\phi) & \cos(\phi) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Rotationen sker medurs sett i axelriktningarna.

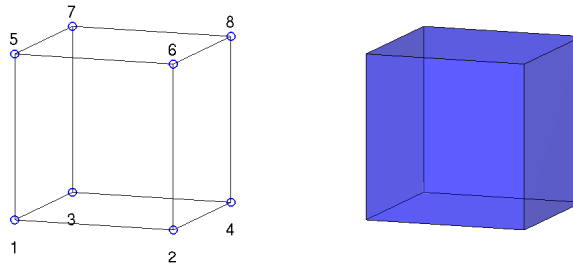
Motsvarande för skalning och translation blir

$$\mathbf{S}(\mathbf{x}) = \mathbf{B}\mathbf{x} = \begin{bmatrix} s_1 & 0 & 0 \\ 0 & s_2 & 0 \\ 0 & 0 & s_3 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}$$

respektive

$$\mathbf{F}(\mathbf{x}) = \mathbf{x} + \mathbf{t} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} + \begin{bmatrix} t_1 \\ t_2 \\ t_3 \end{bmatrix}$$

Nu skall ni rita en tetraeder som ni sedan skall transformera på lite olika sätt. För att underlätta ritandet av tetraedern visar vi hur man kan rita en kub.



Vi ritar en enhetskub enligt

```
H=[0 1 0 1 0 1 0 1 % H(:,j), j:te kolonnen i H, ger koordinater för hörnpunkt j
    0 0 1 1 0 0 1 1 % size(H,2) ger antal hörnpunkter
    0 0 0 0 1 1 1 1];
S=[1 2 4 3 % S(i,:), i:te raden i S, ger nr på hörnpunkter på sida i
   1 2 6 5
   1 3 7 5
   3 4 8 7
   2 4 8 6
   5 6 8 7]; % size(S,1) ger antal sidor
figure(1), clf, hold on
for i=1:size(S,1)
    Si=S(i,:); fill3(H(1,Si),H(2,Si),H(3,Si),'b','facealpha',0.7)
end
axis equal, axis tight, axis off, hold off, view(20,10)
```

Lägg lite tid på att tänka igenom det vi gjort. Hörnpunkternas koordinater ligger som kolonner i matrisen H. På raderna i matrisen S har vi numren på hörnpunkterna på sidorna, t.ex. första raden (1 2 4 3) ger numren på hörnpunkterna som ger botten på kuben.

Antal kolonner i H, som ges av `size(H,2)`, är antalet hörnpunkter. Antalet rader i S, som ges av `size(S,1)`, är antalet sidor.

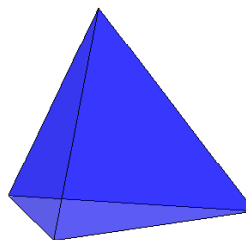
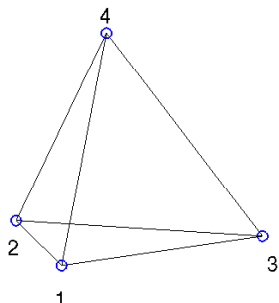
Med `for`-satsen ritas vi upp alla sidor. Vi plockar ut en sidas hörnpunkter med `Si=S(i,:)` och motsvarande kolonner i H, dvs. `H(:,Si)` ger koordinaterna för hörnpunkterna.

Vi ritas sidan med `fill3`, som fungerar som `fill` fast i rummet. Här måste vi separera x_1 -, x_2 - och x_3 -koordinaterna. Med `H(1,Si)` får vi x_1 -koordinaterna för hörnpunkterna på sidan, osv.

Vi får kuben lite lätt genomskinlig med `'facealpha',0.7`. För solid kub sätt `'facealpha',1` eller utelämnas. Med `axis equal` får vi skalor på axlarna så att en kub ser ut som en kub och inte

blir tillplattad. Vidare ger `axis tight` ett koordinatsystem *utan* luft runt kuben som vi ritat och `axis off` gör att axlarna och skalmarkeringar inte syns. Med `view(20,10)` ges betraktelsevinklar, se gärna hjälptexterna.

Uppgift 2. Rita en liksidig tetraeder.



Vi kan ta hörnpunkter på enhetssfären med hörnpunkternas koordinater som

$$\left(\frac{2\sqrt{2}}{3}, 0, -\frac{1}{3}\right), \quad \left(-\frac{\sqrt{2}}{3}, \pm\sqrt{\frac{2}{3}}, -\frac{1}{3}\right), \quad (0, 0, 1)$$

Uppgift 3. Roterat tetraedern runt någon axel. Gör en translation av tetraedern bort från origo. Roterat den åter runt någon axel. Välj ett koordinatsystem så att tetraedern syns i alla lägen den hamnar i.

4 Ortogonal projektion och spegling

Ortogonal eller vinkelrät projektion av en punkt $\mathbf{x}$ på linjen

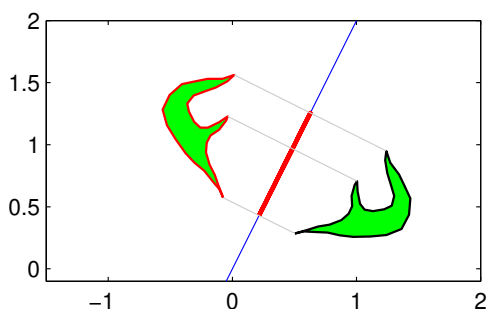
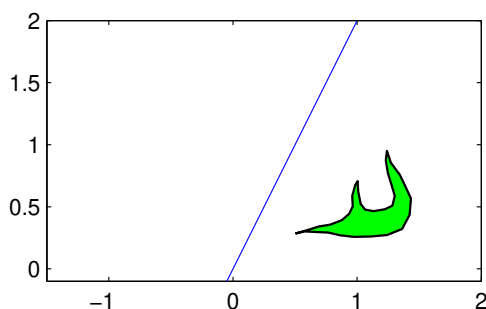
$$ax + by = d$$

ges av $\hat{\mathbf{x}} = \mathbf{x} + \alpha \mathbf{n}$, där $\mathbf{n} = (a, b)$ är normalen och

$$\alpha = \frac{d - \mathbf{n} \cdot \mathbf{x}}{\mathbf{n} \cdot \mathbf{n}}$$

Speglingen i linjen av $\mathbf{x}$ ges av $\mathbf{x}_r = \mathbf{x} + 2\alpha \mathbf{n}$.

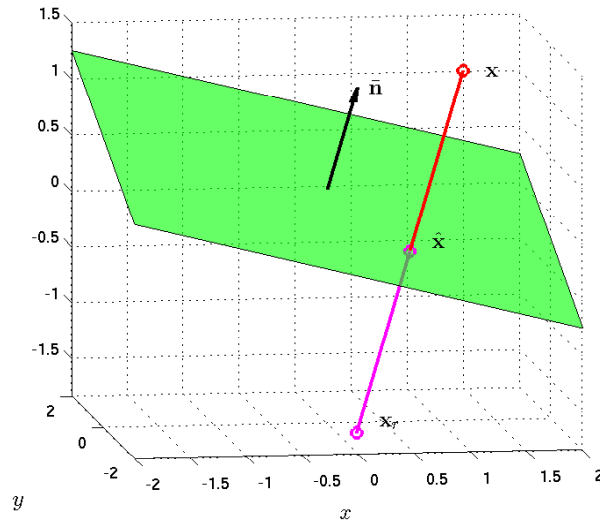
Om vi projicerar och speglar vårt område i en rät linje skulle det kunna se ut så här.



Nu skall vi se hur det blir i rummet: Bestäm ortogonala projektionen på planet

$$ax + by + cz = d$$

Planet har normalvektorn $\mathbf{n} = (a, b, c)$. I bilden har vi ritat enhetsnormal $\bar{\mathbf{n}}$ och ortogonala projektionen $\hat{\mathbf{x}}$ längs enhetsnormalen av en punkt $\mathbf{x}$.



Vi gör ansatsen $\hat{\mathbf{x}} = \mathbf{x} + \alpha \mathbf{n}$, där α skall bestämmas så att $\hat{\mathbf{x}}$ ligger på planet. Ekvationen för planet kan skrivas

$$\mathbf{n} \cdot \mathbf{x} = d$$

och sätter vi in ansatsen får vi

$$\mathbf{n} \cdot \hat{\mathbf{x}} = \mathbf{n} \cdot (\mathbf{x} + \alpha \mathbf{n}) = \mathbf{n} \cdot \mathbf{x} + \alpha \mathbf{n} \cdot \mathbf{n} = d$$

och därmed

$$\alpha = \frac{d - \mathbf{n} \cdot \mathbf{x}}{\mathbf{n} \cdot \mathbf{n}}$$

För speglingen av $\mathbf{x}_r$ av punkten $\mathbf{x}$ i planet gäller

$$\mathbf{x}_r = \mathbf{x} + 2\alpha \mathbf{n}$$

med samma val av α .

Nu skall vi i MATLAB rita planet $ax + by + cz = d$, för $a = 1, b = -1, c = 4$ och $d = 1$.

Eftersom $c \neq 0$ så kan vi lösa ut z , i annat fall får vi modifiera koden

```
>> xmin=-2; xmax=2; ymin=-2; ymax=2;
>> a=1; b=-1; c=4; d=1;
>> X=[xmin xmax xmax xmin]; Y=[ymin ymin ymax ymax];
>> Z=(d-a*X-b*Y)/c;
>> figure(1), clf
>> fill3(X,Y,Z,'g','facealpha',0.7)
>> xlabel('x'), ylabel('y')
>> grid on
```

Resultatet blir planet i figuren ovan.

Uppgift 4. Rita planet vi just tittade på. Bestäm en normalvektor och rita ut den med en pil från en punkt på planet. En pil i rummet kan vi rita med `quiver3(x,y,z,a,b,c,s)`, där x, y, z ger koordinaterna för den punkt som pilen skall ritas från, a, b, c ger pilens utsträckning och s är en skalfaktor (normalt tar vi $s = 0$ vilket ritar utan skalning, medan t.ex. $s = 2$ gör pilarna dubbelt så långa). Välj en punkt $\mathbf{x}$, rita ut den, bestäm dess vinkelräta projektion $\hat{\mathbf{x}}$ på planet och rita ut även den. Slutligen rita också ut speglingen $\mathbf{x}_r$ av $\mathbf{x}$. Markera normalvektorn och de olika punkterna med texter. T.ex. normalvektorn kan markeras med `text(u,v,w,'n')`, där u, v, w är koordinaterna för positionen av vänster sida av texten och `'n'` är texten som skall skrivas, dvs. ett n . Använd `axis equal` så att vi ser om vinklar är räta.

Vi skall se lite mer på den ortogonala projektionen på ett plan $ax + by + cz = d$ där a, b, c och d är konstanter.

Om $\mathbf{x}$ är den punkt vi projicerar och $\hat{\mathbf{x}}$ är projektionen så gäller

$$\hat{\mathbf{x}} = \mathbf{x} + \alpha \mathbf{n}, \quad \alpha = \frac{d - \mathbf{n} \cdot \mathbf{x}}{\mathbf{n} \cdot \mathbf{n}}$$

där $\mathbf{n} = (a, b, c)$ är en normalvektor till planet.

Om $d = 0$, dvs. planet går genom origo, har vi

$$\hat{\mathbf{x}} = \mathbf{x} + \alpha \mathbf{n} = \mathbf{x} - \frac{\mathbf{n} \cdot \mathbf{x}}{\mathbf{n} \cdot \mathbf{n}} \mathbf{n}$$

Eftersom $\mathbf{n} \cdot \mathbf{x}$ är en skalär så gäller

$$\hat{\mathbf{x}} = \mathbf{x} - \frac{(\mathbf{n} \cdot \mathbf{x})}{\mathbf{n} \cdot \mathbf{n}} \mathbf{n} = \mathbf{x} - \frac{1}{\mathbf{n} \cdot \mathbf{n}} \mathbf{n} (\mathbf{n} \cdot \mathbf{x})$$

Vidare gäller att $\mathbf{n} \cdot \mathbf{x} = \mathbf{n}^T \mathbf{x}$, dvs. skalärprodukten kan beräknas genom att $\mathbf{n}^T$ som är en radvektor matrismultipliceras med kolonnvektorn $\mathbf{x}$ och vi får

$$\hat{\mathbf{x}} = \mathbf{x} - \frac{1}{\mathbf{n} \cdot \mathbf{n}} \mathbf{n} (\mathbf{n}^T \mathbf{x}) = \mathbf{x} - \frac{1}{\mathbf{n} \cdot \mathbf{n}} (\mathbf{n} \mathbf{n}^T) \mathbf{x} = \left(\mathbf{I} - \frac{1}{\mathbf{n} \cdot \mathbf{n}} \mathbf{n} \mathbf{n}^T \right) \mathbf{x}$$

där vi utnyttjade att $\mathbf{n} (\mathbf{n}^T \mathbf{x}) = (\mathbf{n} \mathbf{n}^T) \mathbf{x}$.

Observera att $\mathbf{n} \mathbf{n}^T$ är en matris (en s.k. ytterprodukt), medan $\mathbf{n} \cdot \mathbf{n} = \mathbf{n}^T \mathbf{n}$ är ett tal (skalärprodukt eller innerprodukt).

Vi har kommit fram till

$$\hat{\mathbf{x}} = \left(\mathbf{I} - \frac{1}{\mathbf{n} \cdot \mathbf{n}} \mathbf{n} \mathbf{n}^T \right) \mathbf{x} = \mathbf{P} \mathbf{x}$$

Alltså en linjär avbildning med standardmatrisen $\mathbf{P}$.

För spegling gäller

$$\mathbf{x}_r = \left(\mathbf{I} - \frac{2}{\mathbf{n} \cdot \mathbf{n}} \mathbf{n} \mathbf{n}^T \right) \mathbf{x} = \mathbf{R} \mathbf{x}$$

Alltså en linjär avbildning med standardmatrisen $\mathbf{R}$.

Om $d \neq 0$, dvs. planet går *inte* genom origo, har vi

$$\hat{\mathbf{x}} = \mathbf{x} + \alpha \mathbf{n} = \mathbf{x} + \frac{d - \mathbf{n} \cdot \mathbf{x}}{\mathbf{n} \cdot \mathbf{n}} \mathbf{n} = \left(\mathbf{I} - \frac{1}{\mathbf{n} \cdot \mathbf{n}} \mathbf{n} \mathbf{n}^T \right) \mathbf{x} + \frac{d}{\mathbf{n} \cdot \mathbf{n}} \mathbf{n} = \mathbf{P} \mathbf{x} + \beta \mathbf{n}$$

dvs. inte en linjär avbildning. Motsvarande gäller även för spegling, då $d \neq 0$.